HPC: HW1 Report

2017-19841 최창민

1. Compilation Process

1.1 Preprocessing

(a) stdio.h, math.h의 위치와 라인수

```
shpc145@login0:~/hw1$ cpp -v /dev/null
Using built-in specs.
COLLECT_GCC=cpp
OFFLOAD_TARGET_NAMES=nvptx-none
OFFLOAD_TARGET_DEFAULT=1
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 7.5.0-3ubuntu1~18.04' --wit
efix=x86_64-linux-gnu- --enable-shared --enable-linker-build-id --libexecdir=/usr/lib --wi
libstdcxx-abi=new --enable-gnu-unique-object --disable-vtable-verify --enable-libmpx --ena
-list=m32,m64,mx32 --enable-multilib --with-tune=generic --enable-offload-targets=nvptx-no
Thread model: posix
gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)
COLLECT_GCC_OPTIONS='-E' '-v' '-mtune=generic' '-march=x86-64'
 /usr/lib/gcc/x86_64-linux-gnu/7/cc1 -E -quiet -v -imultiarch x86_64-linux-gnu /dev/null -
ignoring nonexistent directory "/usr/local/include/x86_64-linux-gnu"
ignoring nonexistent directory "/usr/lib/gcc/x86_64-linux-gnu/7/../../../../x86_64-linux-g
#include "..." search starts here:
#include <...> search starts here:
 /usr/lib/gcc/x86_64-linux-gnu/7/include
 /usr/local/include
 /usr/lib/gcc/x86_64-linux-gnu/7/include-fixed
 /usr/include/x86_64-linux-gnu
 /usr/include
End of search list.
# 1 "/dev/null"
# 1 "<built-in>"
# 1 "<command-line>"
# 31 "<command-line>"
# 1 "/usr/include/stdc-predef.h" 1 3 4
# 32 "<command-line>" 2
# 1 "/dev/null"
COMPILER_PATH=/usr/lib/gcc/x86_64-linux-gnu/7/:/usr/lib/gcc/x86_64-linux-gnu/7/:/usr/lib/g
LIBRARY_PATH=/usr/lib/gcc/x86_64-linux-gnu/7/:/usr/lib/gcc/x86_64-linux-gnu/7/../../../x86
:/lib/:/usr/lib/
COLLECT_GCC_OPTIONS='-E' '-v' '-mtune=generic' '-march=x86-64'
```

이 출력 결과물을 보면 알 수 있지만, #include의 경우 다음과 같은 순서로 서칭합니다.

1. **/usr/lib/gcc/x86_64-linux-gnu/7/include**
2. **/usr/local/include**
3. **/usr/lib/gcc/x86_64-linux-gnu/7/include-fixed**
4. **/usr/include/x86_64-linux-gnu**
5. **/usr/include**

모두 /usr 아래에 있기 때문에, find /usr -name filename 같은 방식으로 위치를 찾고, wc -l /path/to/file 의 방식으로 줄 수를 세었습니다.

stdio.h

```
shpc1450login0:~/hw1$ find /usr -name stdio.h
/usr/local/cuda-11.4/targets/x86_64-linux/include/cuda/std/detail/libcxx/include/stdio.h
/usr/include/stdio.h
/usr/include/c++/7/tr1/stdio.h
/usr/include/x86_64-linux-gnu/bits/stdio.h
```

이 중 해당하는 것은 **/usr/include/stdio.h** 와 **/usr/include/x86_64-linux-gnu/bits/stdio.h** 지만, 후자는 **stdio.h**가 아닌 **bits/stdio.h** 이기 때문에 해당하지 않습니다.
즉, **/usr/include/stdio.h**에 위치하고 있습니다.

```
shpc1450login0:~/hw1$ wc -l /usr/include/stdio.h
870 /usr/include/stdio.h
```

줄 수는 870 라인입니다.

math.h

```
870 /usr/include/stdio.h
shpc1450login0:~/hw1$ find /usr -name math.h
/usr/local/cuda-11.4/targets/x86_64-linux/include/thrust/detail/mpl/math.h
/usr/local/cuda-11.4/targets/x86_64-linux/include/cuda/std/detail/libcxx/include/math.h
/usr/include/c++/7/tr1/math.h
/usr/include/c++/7/math.h
/usr/include/math.h
/usr/include/ucs/sys/math.h
```

이 중 해당하는 것은 아래 네 개가 모두 해당하지만, 마찬가지로 /usr/include의 바로 밑에 있는 것은 5번째 줄이므로 **/usr/include/math.h** 에 위치하고 있습니다.

```
shpc1450login0:~/hw1$ wc -l /usr/include/math.h
1266 /usr/include/math.h
```

줄 수는 1266 라인입니다.

(b) Preprocess 결과물에서 scanf, printf, sqrt 함수

scanf

```
extern int scanf (const char *__restrict __format, ...) ;
```

printf

```
extern int printf (const char *__restrict __format, ...);
```

sqrt

```
  extern double sqrt (double __x) __attribute__ ((__nothrow__ , __leaf__));
extern double __sqrt (double __x) __attribute__ ((__nothrow__ , __leaf__));
```

(c) scanf, printf, sqrt의 실제 구현이 들어있는가

실제 구현이 포함되어 있지 않음. 이는 실제 구현은 작성한 코드 외부의 라이브러리에 포함되어 있고, 이후 linking 과정에서 이들이 우리가 작성한 코드와 합쳐진다

1.2 Compilation

(a) Object file을 출력하는 gcc 커맨드

gcc -c sqrt.c 임. **man gcc** 를 통해서 찾을 수 있었음

(b) sqrt.o의 파일 포맷

```
shpc1450@login0:~/hw1$ file sqrt.o
sqrt.o: ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV), not stripped
shpc1450@login0:~/hw1$ file sqrt.c
sqrt.c: C source, ASCII text
shpc1450@login0:~/hw1$
```

file sqrt.o 명령을 통해 찾았으며 ELF relocatable 파일이다.

1.3 Linking

(a) 컴파일 에러의 이유와 이걸 해결할 수 있는 커맨드

에러가 발생하는 이유는 라이브러리의 실제 구현이 sqrt.o에 없기 때문이다. 실제 구현인 library를 linking해주기 위해서는 gcc -o sqrt sqrt.o -lm 를 하면 된다. libc는 자동으로 linking이 되고 math라이브러리는 명시적으로 -lm을 붙여주어 linking한다.

(b) 임의의 수를 입력한 결과

```
shpc145@login0:~/hw1$ ./sqrt
Usage: ./sqrt number
Example: ./sqrt 2
shpc145@login0:~/hw1$ ./sqrt 3
1.73205081
shpc145@login0:~/hw1$ ./sqrt 4
2.00000000
shpc145@login0:~/hw1$ ./sqrt 100
10.00000000
shpc145@login0:~/hw1$ ./sqrt 100123
316.42218633
shpc145@login0:~/hw1$ ./sqrt 129387542
11374.86448271
shpc145@login0:~/hw1$ ./sqrt 0
0.00000000
shpc145@login0:~/hw1$ ./sqrt -2
-nan
```

3. 클러스터 사용 연습

(a) sinfo

```
shpc145@login0:~/hw1$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
shpc         up       1:00      4   idle a[08-11]
```

sinfo는 compute node들의 상태를 보여줍니다. 각각의 항목들은 다음 뜻을 가집니다

- PARTITION: compute node들의 그룹인 partition의 이름을 나타냅니다.
- AVAIL: 해당 partition이 사용 가능한 지 표시합니다.
- TIMELIMIT: 분:초 로 표시되며 compute node에서 얼마나 오래 돌릴 수 있는 지를 표시합니다.
- NODES: 현재 partition에 소속되어 있는 node의 수입니다.
- STATE: compute node들의 상태입니다. idle하거나 mix등등의 상태를 가집니다.
- NODELIST: compute node들의 이름입니다.

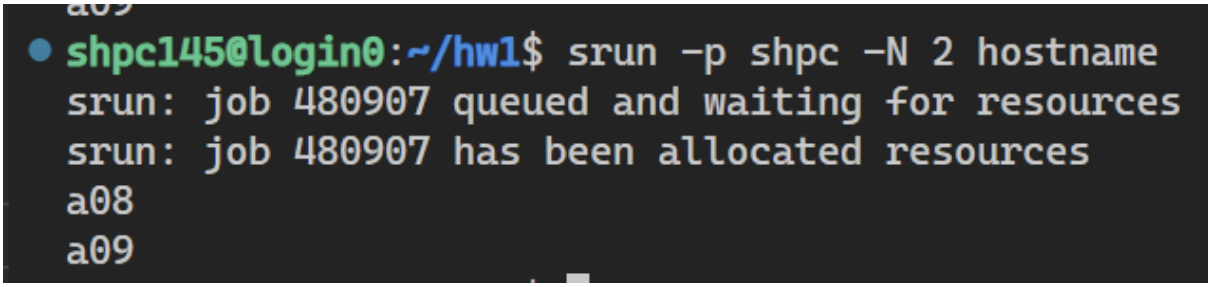
(b) squeue

```
shpc145@login0:~/hw1$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
shpc145@login0:~/hw1$
```

squeue는 제출된 job들의 상태를 보여줍니다.

- JOBID: 제출된 job의 id
- PARTITION: compute node들의 그룹인 partition의 이름
- NAME: job이나 job step의 이름입니다.
- USER: 제출한 사람의 이름
- ST: job의 상태코드입니다. 대표적으로 PD는 펜딩, R는 running, CG는 compiling입니다.
- TIME: 문서를 봤을 때 실행 이후 걸린 시간으로 추정됩니다.
- NODES: job에 할당된 node입니다.
- NODELIST: job에 할당된 nodelist입니다.

(c) srun



해당 명령어는 shpc파티션에 2개의 노드를 할당받아서 hostname이라는 명령을 실행하는 것입니다. 이에 따라서 hostname이 출력된 것을 알 수 있습니다.

(d) lscpu

로그인 노드

```
shpc145@login0:~$ lscpu
Architecture:        x86_64
CPU op-mode(s):      32-bit, 64-bit
Byte Order:          Little Endian
CPU(s):              128
On-line CPU(s) list: 0-127
Thread(s) per core:  2
Core(s) per socket:  32
Socket(s):           2
NUMA node(s):        2
Vendor ID:           AuthenticAMD
CPU family:          23
Model:               49
Model name:          AMD EPYC 7502 32-Core Processor
Stepping:            0
CPU MHz:             1500.009
CPU max MHz:         2500.0000
CPU min MHz:         1500.0000
BogoMIPS:            5000.07
Virtualization:      AMD-V
L1d cache:           32K
L1i cache:           32K
L2 cache:            512K
L3 cache:            16384K
NUMA node0 CPU(s):   0-31,64-95
NUMA node1 CPU(s):   32-63,96-127
Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ht
 syscall nx mmxext fxsr_opt pdpe1gb rdtscp lm constant_tsc rep_good nopl xtopology nonstop_tsc cpuid extd_apicid aperfmpe
rf pni pclmulqdq monitor ssse3 fma cx16 sse4_1 sse4_2 movbe popcnt aes xsave avx f16c rdrand lahf_lm cmp_legacy svm extap
ic cr8_legacy abm sse4a misalignsse 3dnowprefetch osvw ibs skinit wdt tce topoext perfctr_core perfctr_nb bpext perfctr_l
lc mwaitx cpb cat_l3 cdp_l3 hw_pstate sme ssbd ibrs ibpb stibp vmmcall fsgsbase bmi1 avx2 smep bmi2 cqm rdt_a rdseed adx
smap clflushopt clwb sha_ni xsaveopt xsavec xgetbv1 xsaves cqm_llc cqm_occup_llc cqm_mbm_total cqm_mbm_local clzero irper
f xsaveerptr arat npt lbrv svm_lock nrip_save tsc_scale vmcb_clean flushbyasid decodeassists pausefilter pfthreshold avic
 v_vmsave_vmload vgif umip rdpid overflow_recov succor smca
shpc145@login0:~$
```

srun -p shpc -N 1 lscpu

```
shpc145@login0:~$ srun -p shpc -N 1 lscpu
srun: job 480908 queued and waiting for resources
srun: job 480908 has been allocated resources
Architecture:        x86_64
CPU op-mode(s):      32-bit, 64-bit
Byte Order:          Little Endian
CPU(s):              64
On-line CPU(s) list: 0-63
Thread(s) per core:  2
Core(s) per socket:  16
Socket(s):           2
NUMA node(s):        2
Vendor ID:           GenuineIntel
CPU family:          6
Model:               85
Model name:          Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz
Stepping:            7
CPU MHz:             1855.079
CPU max MHz:         2101.0000
CPU min MHz:         800.0000
BogoMIPS:            4200.00
Virtualization:      VT-x
L1d cache:           32K
L1i cache:           32K
L2 cache:            1024K
L3 cache:            22528K
NUMA node0 CPU(s):   0-15,32-47
NUMA node1 CPU(s):   16-31,48-63
Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr ss
e sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_t
sc cpuid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid dca sse4_1 sse
4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb cat_l3 cdp
_l3 invpcid_single intel_ppin ssbd mba ibrs ibpb stibp ibrs_enhanced tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_a
djust bmi1 hle avx2 smep bmi2 erms invpcid rtm cqm mpx rdt_a avx512f avx512dq rdseed adx smap clflushopt clwb intel_pt av
x512cd avx512bw avx512vl xsaveopt xsavec xgetbv1 xsaves cqm_llc cqm_occup_llc cqm_mbm_total cqm_mbm_local dtherm ida arat
 pln pts pku ospke avx512_vnni md_clear flush_l1d arch_capabilities
```

의미

- 첫번째 명령은 로그인 노드에서 **lscpu**를 한 것이므로, 로그인 노드의 cpu정보를 출력합니다. AMD EPYC CPU를 사용하는 것을 알 수 있습니다.
- 두번째 명령은 compute node를 할당받아서 **lscpu**를 한 것이므로 compute node의 cpu정보를 출력합니다. Intel Xeon CPU를 사용하는 것을 알 수 있습니다.
- 두 개의 명령어의 출력이 다른 이유는, 서로 다른 컴퓨터에 있는 서로 다른 CPU의 정보를 출력하게 했기 때문입니다.